Analog Control Types

Channel device control

Fri, Oct 10, 1997

The Local Station/IRM software supports a variety of hardware interfaces. Part of the support is that given to analog channel settings. The information in the analog control field of the analog descriptor for a given channel device describes the specific form of control which is to be used for that channel. This note details the various forms of the analog control field for each type. If the setting action appears successful (no bus error or other errors), the setting word field is updated with the data value for the given channel.

Setting data used for analog control is considered in the range of ordinary two-complement arithmetic, `0x8000`—`0x7FFF`, nominally corresponding to –10 to +10 volts. Analog control support for a specific type of hardware must map values in this range to the format expected by the hardware. Upon successfully performing a setting, the setting word field is updated with the data value in the standard range.

The analog control field consists of 4 bytes. The first byte is the analog control type#, a small index value. The meaning of the other 3 bytes depends upon the type# byte. A few of the types are historic. The reference to "early Linac" dates back to the days of Multibus-based 68000 stations that no longer exist. The relevant code is very short, so including this support is not costly.) The types currently supported are:

`00`	No analog control (parameter page will not mark it with a "–")
`01`	Datel Multibus D/A (obsolete: used in early Linac)
`02`	Motor (setting value is desired reading, relative setting is #steps)
`03`	Bipolar multiplex D/A (hardware used in Linac, now accessed via SRM)
`04`	Unipolar multiplex D/A (hardware used in Linac, now accessed via SRM)
`05`	Memory word (accessed as two bytes)
`06`	Intel 8253 timer (obsolete: used in early Linac)
`07`	M6840 timer (obsolete: used in early Linac)
`08`	1553 D/A (12-bit)—used in rack monitor
`09`	Analog Devices RTI-602 D/A board
`0A`	Memory word (accessed as one word)—also implied for types 40–7F
`0B`	Message queue setting to another cpu (co-processor)
`0C`	Unsigned 12-bit D/A (in short I/O space)
`0D`	Burr-Brown MPV904 12-bit D/A board
`0E`	1553 D/A (16-bit)
`0F`	AMD9513 timer (32-bits from pair of channels)
`10`	Memory byte (single byte no shift)
`11`	Memory byte (single byte w/ shift in short I/O space)
`12`	Target same channel's reading word (w/mask option)
`13`	SRM analog control (12-bit)
`14`	SRM analog control (16-bit)
`15`	IRM D/A (w/offset and 4-bit right shift)
`16`	IRM D/A (w/offset but w/o right shift)
`17`	Unipolar 15-bit D/A clamped at zero
`18`	Unsigned 16-bit D/A or timer
`19`	PLC memory word via `PLCQ` message queue
`1A`	Greensprings IndustryPack D/A board
`1B`	IndustryPack timer board trigger event select/clear

`40–7F` Memory word (type `0A`) using all 4 bytes for the 32-bit address.

Analog control field formats

`00 xx xxxx`
No analog control (parameter page will not mark it with a "–"). Last 3 bytes may be used to hold memory address for another purpose.

`01 c# addr`
Datel Multibus D/A (obsolete: used in early Linac)
The `c#` byte is the board's channel#. The `addr` word is the board's address (sign-extended). No such hardware is used in VME systems.

`02 xx xxxx`
Motor control—all forms
Motors do not have a setting value, but they can be set to a value, in which case one of the scale factor constants is used to derive the required #steps to be issued to the motor interface according to the difference between the present reading and the desired reading. The constant used is that normally used for the D/A full scale value, and it has the value of the number of motor steps to effect a 10 volt, or full scale, change in the reading. No iterating logic is used; only one try is made. Motors can also be sent a delta value, either in raw units or in engineering units. Finally, they can be sent a number of steps as the data value. these cases are distinguished by the listype# used in the setting command.

Listype	*#bytes*	*Data*
1	2	desired reading
7	2	#steps
39	2	delta reading
41	4	desired engineering units reading
44	4	delta engineering units reading

`02 0b addr`
`02 1b addr`
Memory-mapped motor
The `b` is the bit# in the range 0–7 for the byte addressed by the 16-bit address in VME short I/O addressing space. (For `b` in range `8–F`, use Vertical Interconnect crate `0x10` short I/O space, as used in D0.) Motor pulses (~20 microsec hi-active pulses) are formed at an interrupt-driven 150 Hz rate—or another rate specified in microsec units by the system parameter found at byte offset `0x14` in the first entry of the PAGEM table. The difference between the `0b` case and the `1b` case are the same as described below under the `Eb` and `Fb` cases.

`02 cBlkPt`
1553-based motor
The `cBlkPt` field is a 3-byte ptr to the 1553 command block structure. It must be in the range `0x200000–0xA0FFFF`. (For the 1553 support in the D0 local stations, the 1553 controller memories are accessed at `0x2xxxxx`, so that the second byte in the analog control field is always `0x2x`.) The command block houses the 1553 command for a single word write of the two's-complement #steps to be issued to the motor. The external hardware is expected to deliver the pulses. The 150 Hz, or whatever, interrupt that is used for the memory-mapped case above decrements a counter in the analog data table entry for the same channel that carries this analog control entry in order to provide a shadow of the countdown register. This

can be used to determine whether the motor is still running, if it can assume that the external hardware's motor pulse rate is the same rate.

```
02    Ax    xxxx
```

SRM-based motor

The `Ax` byte specifies the arcnet node address of the SRM being targeted. A short message is formed and transmitted via arcnet to that SRM, which then performs the actual setting. The setting is presumed successful if it passes all checks prior to transmitting the arcnet message. As always, the absolute guarantee that a setting was successful can only be determined by checking that the desired change actually occurred. The `xxxx` word specifies the parameters necessary for the SRM's needs. See the document *SRM Message Protocols* for more details on this.

```
02    Eb    byt#
02    Fb    byt#
```

Indirectly-addressed motor

The target address is derived from the `byt#` entry in the BADDR table of Binary Addresses that is accessed for generic digital status/control. This provides for using an arbitrary 32-bit target address.

The difference between the `Eb` case and the `Fb` case is the type of motor interface. The `Eb` case is a CW-CCW pulse case, whereas the `Fb` case is a pulse and direction control case. For the CW/CCW case, the indicated bit# of the target byte specifies the control line to be pulsed for CW steps. The bit# to be used for CCW pulses is then the adjacent bit#. There are 4 adjacent pairs of bits that can be used in one byte: 0/1, 2/3, 4/5, and 6/7. In addition, there is a choice about the active polarity of the pulse. The least significant bit of the bit# specifies the active state of the pulse. If the bit# is 1/3/5/7, the active state is a one, so the pulses will be low-going-high pulses. If the bit# is 0/2/4/6, the active state is a zero, so the pulses will be high-going-low pulses. In all cases, the length of the pulse is about 20 microsec, allowing for some heavy filtering of noise on the control lines, if desired.

The `Fb` case is the pulse control and pulse direction case. In addition, a check of the appropriate limit switch is done before issuing a pulse. There are two VME boards, one made by BurrBrown and another made by Ironics, that work this way. Each motor uses two control lines and two limit switch status lines, so that one control byte and one status byte can interface up to 4 motors. The difference between the two boards is the separation of the addressing of the control byte with the status byte. The Ironics logic places the status byte two bytes after the control byte, whereas the BurrBrown board places the status byte in the byte following the control byte. (The Ironics board control byte address is odd, and the BurrBrown control byte address is even, so that the code can determine which logic to follow.) The CW direction state is zero; the CCW direction state is one. The bit# specified is that used for the pulse control line; the adjacent bit of the pair is used for direction control. The CW limit switch is in the bit position of the status byte that matches the pulse control bit#. The CCW status bit is in the adjacent status bit. When a status bit is zero, it means the motor-driven device is off limit, so that pulsing is permitted; when the status bit is a one, no pulses will be issued. The pulse control active state is specified by the least significant bit of the bit# of the pulse control line. (If one gets the active state of the pulse control wrong, it may have to switch the two control and status bits used.)

```
03    c#    addr
```

Bipolar multiplex D/A (hardware used in Linac, now accessed via SRM)

The c# byte includes the chassis# 0–7 and the channel# 0–15 concatenated to form a 7-bit value. If the sign bit of the c# byte is set, the addr word is prefixed with 0x10FF to form a target address via Vertical Interconnect rate 0x10 short I/O space; otherwise, the addr word is prefixed with 0xFFFF to place the target address in the VME short I/O space.

04 c# addr

Unipolar multiplex D/A (hardware used in Linac, now accessed via SRM)

This is the same as type#3, but the setting value is clamped to zero if negative.

05 memPtr

Memory word (accessed as two bytes)

The memPtr 24-bit memory address is mapped into a 32-bit address as follows:

If the address is < 0xF00000, the hi byte of the 32-bit address is 0x00.

If the address is ≥ 0xF00000, the hi byte of the 32-bit address is 0xFF.

This mapping scheme allows entry of short I/O addresses plus all addresses below 15 Mbytes. The data word is written as two consecutive bytes. (See type 0A for writing the data word as a 16-bit word.)

06 c# addr

Intel 8253 timer (obsolete: used in early Linac)

The c# byte is in the range 0–2. The addr is sign-extended to form the address of the 8253 chip. (Not used in VME system)

07 xx addr

Motorola 6840 timer (obsolete: used in early Linac)

The addr ends in 2, 4 or 6 to indicate which of three 16-bit timer channels is to be set. Setting values ≤ 0 are clamped to $0001. (Not used in VME system)

08 cBlkPt

1553 D/A (12-bit)—used in rack monitor

The cBlkPt is a ptr to a 1553 command block in the controller's memory. The command block houses the 1553 command for a single word write to a D/A. For the 1553 Rack Monitor controllers used in D0, the ptr is 0x2xxxxx.

09 daAddr

Analog Devices RTI-602 D/A board

The address is mapped into 32-bits via the type#05 scheme. The data value is converted to offset binary and written to the resultant address.

0A memPtr

Memory word (accessed as one word)

The address is mapped into 32-bits via the type#05 scheme. The data word is written as one 16-bit word. (Some hardware boards require this.)

0B ty indx

Message queue setting to another cpu (co-processor)

An analog control message is placed into a co-processor message queue. The cpu# is included in bits 6–4 (mask=0x70) of the type byte. The message placed into the queue is formatted as four words in this way:

```
0008
(ty & $8F)
index
data
```

The first word is the size of the message, which in this case is always 8 bytes.
The second word is the type byte AND-ed with `0x8F` to remove the cpu#. The third word is an `index` which may have any value and serves, in conjunction with the type value, to identify what is controlled to the co-processor cpu. Now the message queue to a co-processor is more general than this use for analog control. Other message types may be passed to the cpu by the use of setting commands that use listype #45. The type word used in those messages should not conflict with those used here. (One should not use listype #45 to send the same message as is used for analog control, because the setting word associated with the analog channel will not get updated.) An easy way to insure there is no conflict is to use type word values ≥ `0x100`, since these analog control messages use index words ≤ `0x8F`.

`0C xx addr`
Unsigned 12-bit D/A (in short I/O space)
The `addr` is assumed to be in short I/O space. The `data` word is clamped to zero if negative.

`0D daAddr`
Burr-Brown MPV904 12-bit D/A board
The address is mapped into 32-bits by the same scheme used in type `#05`. The data value is converted to complement offset binary and right-justified to match what the board expects.

`0E cBlkPt`
1553 D/A (16-bit)
This is the same as type `#08`, but knob control sensitivity (as used with listype#7) is based upon 16 bits rather than 12.

`0 F co addr`
AMD9513 timer (32-bits from pair of channels)
The addr is taken to be in short I/O space. The code values are of 4 types:
`0x` Coarse channel
`1x` Fine channel
`2x` Clock event selection
`3x` Clear all events
The `x` nibble is used to identify the timer channel (0–7 range) that is to be controlled on the clock timer board. More details on this can be found in the document entitled “VME Clock Timer Board.”

`10 memPtr`
Memory byte (single byte no shift)
The lo byte of the data word is written to the given address (mapped to 32 bits using the type `#05` scheme).

`11 sh addr`
Memory byte (single byte w/ shift in short I/O space)

The shift count is used to right shift the data word before writing the lo byte to the given address which is assumed to be in short I/O space.

`12 xx mask`
Same channel reading word w/mask
If the mask=`0000` (nearly always the case) the setting data is copied into both the reading field as well as the setting field. This is a kind of dummy setting, as no hardware is addressed, and the reading field is automatically updated with the most recent setting value. If the mask is nonzero, it indicates the portion of the setting (and reading) field that is settable. The bits in the setting word outside the mask are preserved; only the bits within the mask are affected.

`13 rm ctrl`
SRM setting (12-bit)
The `rm` gives the address of the Smart Rack Monitor, which is in the range `0xA0—BF`. The `ctrl` word encodes the parameters needed by the SRM code to address the D/A. For knob control, 12-bit precision is assumed.

`1 4 rm ctrl`
SRM setting (16-bit)
The `rm` gives the address of the Smart Rack Monitor, which is in the range `0xA0—BF`. The `ctrl` word encodes the parameters needed by the SRM code to address the D/A. For knob control, 16-bit precision is assumed.

`15 daAddr`
IRM D/A with offset and 4-bit right shift
This 12-bit D/A IP board supports offset binary; the data word is increased by `0x8000` and right-shifted 4 bits before writing to the address specified via the type`#05` scheme.

`16 daAddr`
IRM D/A with offset but w/o right shift
This 16-bit D/A IP board uses offset binary; the data word is increased by `0x8000` and written to the address specified via the type`#05` scheme.

`17 daAddr`
Unipolar 15-bit D/A clamped at zero
The setting data word is written to the target address specified via the type`#05` scheme. Any negative setting data word is replaced with zero.

`18 daAddr`
Unsigned 16-bit D/A or timer
The data word is written to the target address (type`#05` scheme) after having adding `0x8000` to it. This method converts a range of `8000—7FFF` into `0000—FFFF`.

`19 ty refA`
PLC memory word via `PLCQ` message queue
The `ty` byte specifies the data type, and the `refA` word specifies the reference address needed for the DirectNET serial protocol used to communicate with the PLC controller. (This was used in the PET project.) See the document DirectNET PLC Access for more information on this.

`1A daAddr`

Greensprings D/A IP board

The setting data word is written to the target address (type`#05` scheme). Then the update bit is set in the board's control register whose address is derived by clearing the low 8 bits of the target address.

`1B cx evtA`

IP timer board trigger event select/clear

This timer board includes a 256-byte array that holds event trigger selections for all 8 timers. The hi-order word of the base address of this array is specified by the `evtA` word. (The lo-order word of the base address is assumed to be `0000`.) The data word specifies in its low byte the clock event# of interest and therefore the byte targeted in the 256-byte array.

The `c` nibble is 0 for clearing an event trigger or 1 for setting an event trigger. The `x` nibble indicates the timer# in the range 0–7.